Project report

Title - Decision tree based classifier model building using ID3 algorithm.

Team details -

1)Sreeja Dacharla (S20200020256)
2)Himaja Anchuri(S20200020264)
3)Dharani Pokuri(S20200020294)
4)Tummepalli Anka Chandrahas Purushotham Gupta(S20200010213)

Date of submission - 24-11-2022

Theory -

- For data preprocessing we need to do three things,
    1. Finding correlation between variables so that highly correlated variables can be removed to reduce the redundancy in the dataset.
        ➔ We used pearson correlation method for numeric data and for the rest we used chi-square correlation method.
    2. Checking whether there are any missing values in the dataset.
        ➔ There are no missing values in the given dataset.
    3. Removing outliers in dataset.

- Decision tree(ID3) -
    ➔ We use decision tree to make a decision that is to predict the class.Each node(other than leaf nodes) is used to make the decision and leaf nodes represent the outcome of the test data.
    ➔ Id3 algorithm is used to get the decision tree of the dataset.
    ➔ In Id3 algorithm,We use Information gain to find the best possible feature.
    ➔ To find information gain,We use entropy(measure of disorder in the target feature of the dataset) using formula -

    $$S=\Sigma(-p_i*\log_2(p_i)) \text{ for } i = 1 \text{ to } n$$

    Where n is number of classes and p is probability of class i.
    ➔ For information gain ,

    $$IG(S, A) = Entropy(S) - \sum((|S_v| / |S|) * Entropy(S_v))$$

    Where Sv is the set of rows in S(dataset)
    ➔ Now split the dataset using the feature with max information gain and if all rows belong to the same class,make the current node as leaf node with the class as its label.Repeat this for the remaining features until we run out of all features or the tree has all leaf nodes.

- Bootstrap sampling method -
    ➔ It estimates the sampling distribution by taking multiple samples with replacement from a single random sample.This is done repeatedly and we will find the average of all the accuracy of samples.

- K - fold cross validation -
  - ➔ In this we take k number of folds(mostly 5/10) and one will be the test set and the rest will be training set.
  - ➔ We will apply the model repeatedly for k times because we need to take every fold as test set for once.
  - ➔ Now we can calculate statistics like accuracy etc.., in each iteration and we will take average of all the iterations.we will decide the best k value based on the maximum statistics.

- Performance measures -
  - ➔ Confusion matrix - It is a matrix of size 2X2 with actual values on one axis and predicted values on the other axis.

	C1	C2
C1	True positive	False negative
C2	False positive	True negative

    Lets say ,
    TN - true negative(both actual and predicted are -)
    TP - true positive(both actual and predicted are +)
    FP - false positive (Actually + but incorrectly classified as -)
    FN - false negative(Actually - but incorrectly classified as +)
  - ➔ Accuracy : Number of instances that are correctly classified i.e.., Accuracy = (TP+TN)/total
  - ➔ Precision:out of + predicted, number of instances that are actually - Precision = TP/(TP+FP)
  - ➔ Recall:out of actually + ,number of instances that are + predicted. Recall = TP/(TP+FN)
  - ➔ F1 score: Harmonic mean of precision and recall. F1 score = 2*Precision*Recall/(Precision+Recall)

Code implementation -

- Installing and loading all the required libraries.

```
install.packages('rpart')
install.packages('caret')
install.packages('rpart.plot')
install.packages('rattle')

#loading the libraries
library(ggplot2)
library(tidyverse)
library(corrplot)
library(dplyr)
library(rpart,quietly = TRUE)
library(caret,quietly = TRUE)
library(rpart.plot,quietly = TRUE)
library(rattle)
library(caret)
```

- Now we read the csv files and we will check whether there are any null values in it so that we can remove it.after that we will remove the unnecessary columns to reduce complexity in dataset.Now we will separate numerical data and categorical data to find correlation.

```
#getting the source data(set the directory to where the data files are in your device)
setwd("C:\\Users\\Chandrahas\\Documents\\IDA-Project-main\\IDA-Project-main")
mydatamat<-read.csv("student-mat.csv",header=TRUE)
mydatapor<-read.csv("student-por.csv",header=TRUE)
mydata=rbind(mydatamat,mydatapor)

#checking for null values
sum(is.na(mydata))

#removing the avoidable columns in dataset to decrease the complexity of the decision tree
mydata=subset(mydata,select=-c(address,famsize,Medu,Fedu,Mjob,
                               Fjob,traveltime,famsup,schoolsup,activities,
                               nursery,reason,paid,guardian))

#separating the numerical and categorical data for correlations
mydata.num=select_if(mydata,is.numeric)
mydata.cat=select_if(mydata,Negate(is.numeric))
```

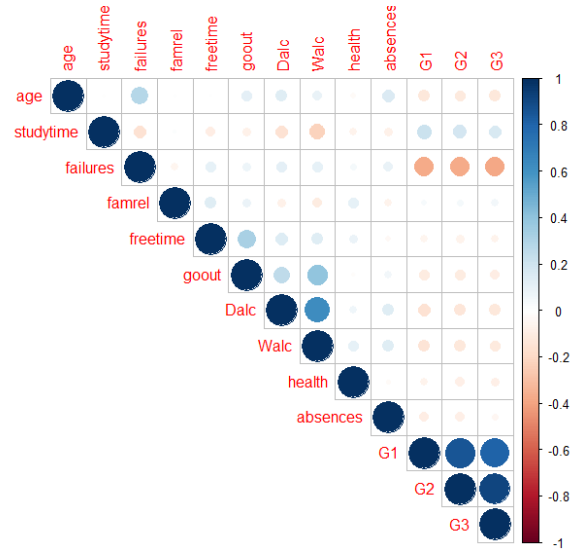
Since there are no null values in dataset we will get as 0

```
> sum(is.na(mydata))
[1] 0
```

- Now we need to find correlation between attributes and if any pair gets their correlation more than 0.9 we will remove it to reduce the redundancy in dataset.
- For numerical data we used pearson correlation and plot output is attached below.

```
#correlation for numerical attributes
cor.mat=cor(mydata.num,method='pearson')
dev.new(width=5,height=5)    #every plot from here will be shown in the dialog box opened just now
corrplot(cor.mat,method='circle',type='upper')

#Checking for certain combinations of attributes from above graph
cor(mydata$Dalc,mydata$walc)
cor(mydata.num$G1,mydata$G2)
cor(mydata.num$G2,mydata$G3)
cor(mydata.num$G3,mydata$G1)

#removing the columns if the correlation is greater than 0.9
mydata=subset(mydata,select=-c(G2))
mydata=subset(mydata,select=-c(G3))
```

Pearson correlation plot for numerical data -



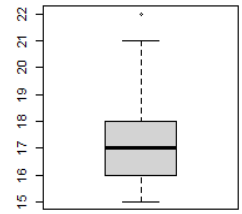
Now the result for the correlation of certain pair of attributes -

```
> cor(mydata$Dalc,mydata$walc)
[1] 0.6278138
> cor(mydata.num$G1,mydata$G2)
[1] 0.8587388
> cor(mydata.num$G2,mydata$G3)
[1] 0.9107432
> cor(mydata.num$G3,mydata$G1)
[1] 0.8091417
```

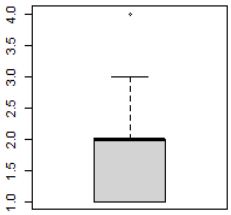
- Now we will remove G2 and G3 since correlation is >0.9 and we will draw box plots for numerical data.

```
mydata=subset(mydata,select=-c(G2))
mydata=subset(mydata,select=-c(G3))

#reassigning the numerical and categorical columns
mydata.num=select_if(mydata,is.numeric)
mydata.cat=select_if(mydata,Negate(is.numeric))

#Box plots for numerical data
num.col=colnames(mydata.num)
par(mfrow=c(2,3))
for (x in 1:6){
  boxplot(mydata.num[,num.col[x]],label=TRUE,xlab=num.col[x])
}

par(mfrow=c(2,3))
for (x in 7:11){
  boxplot(mydata.num[,num.col[x]],label=TRUE,xlab=num.col[x])
}
par(mfrow=c(1,1))
```

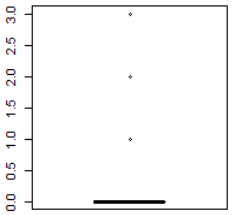
Box plots will be -



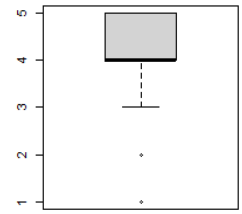
age



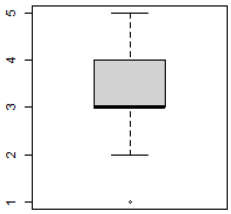
studytime



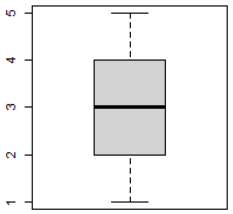
failures



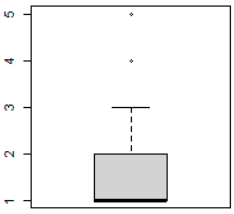
famrel



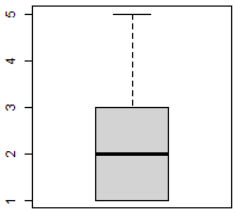
freetime



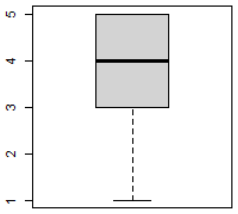
goout



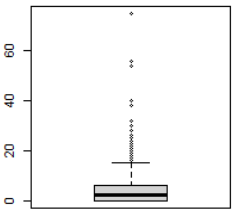
Dalc



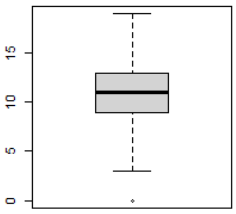
Walc



health



absences



G1

- After preprocessing dimensions of dataset are given below and also we need to find the frequency tables for the first level of ID3.

```
#checking the dimensions of data,before ending the pre-processing
dim(mydata)

#Frequency tables with the class atrribute-Walc for the first level of ID3
col.names=colnames(mydata)
col.names=col.names[col.names %in% "Walc" == FALSE]
for( x in 1:16){
  print(paste('frequency table with:',col.names[x]))
  data=table(mydata$walc,mydata[,x])
  print(data)
}
```

Dimensions of dataset after preprocessing -

```
mydata                  1044 obs. of 17 variables
```

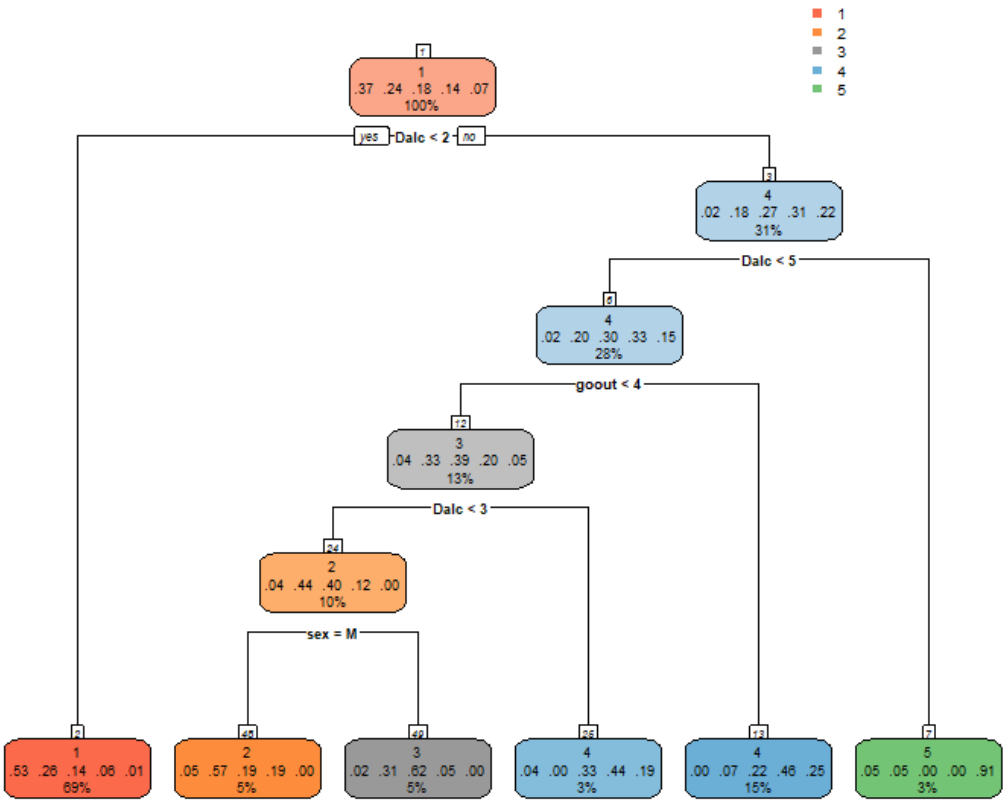
Frequency table for one of the attribute(we will get the rest similarly) -

```
[1] "frequency table with: school"

    GP  MS
1 309  89
2 164  71
3 141  59
4 102  36
5  56  17
```

- Now we will split the dataset into train and test set in ratio 2:1 and we will build the model using rpart.We will also build a confusion matrix.

```
#training the model with standard sampling with 80 percent training data
#the algorithm used is ID3
train <- sample(1:nrow(mydata),size = ceiling(0.80*nrow(mydata)),replace = FALSE)
mydata.train <- mydata[train,]
mydata.test <- mydata[-train,]

model=rpart(walc~., data=mydata.train, method="class")
rpart.plot(model, nn=TRUE)
pred <- predict(model,subset(mydata.test,select=-c(walc)),type="class")
confusionMatrix(factor(pred,levels=1:5),factor(mydata.test$walc,levels=1:5))
```

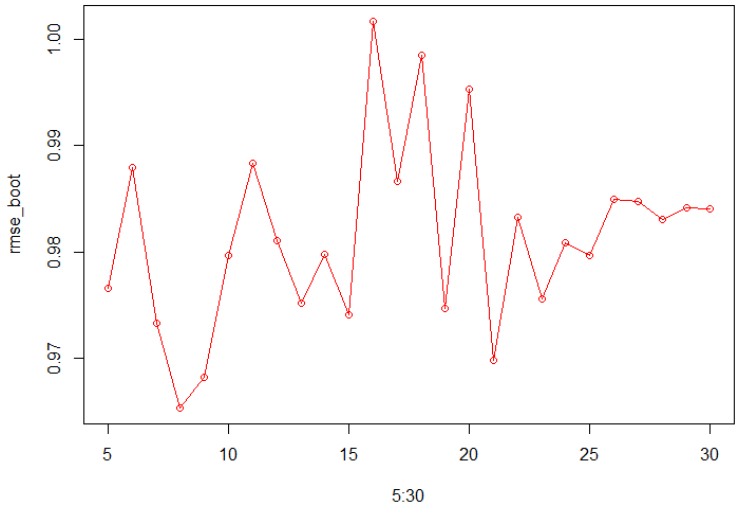
Decision tree will be -



- Now we will do bootstrap resampling for better accuracy.

```
#ID3 with bootstrap resampling
rmse_boot=list()
for (k in 5:30){
  train_control=trainControl(method='boot',number=k,p=0.80)
  #here k is number of times data is resampled
  model=train(Walc~.,data=mydata,method='rpart',trControl=train_control,metric='RMSE',maxdepth=5)
  rmse_boot[[length(rmse_boot)+1]]=model$results$RMSE[1]
  print(model)
}
print(rmse_boot)
plot(5:30, rmse_boot,type="o", col="red")
#the plot is for different number of iteration(number of times it is resampled)
```

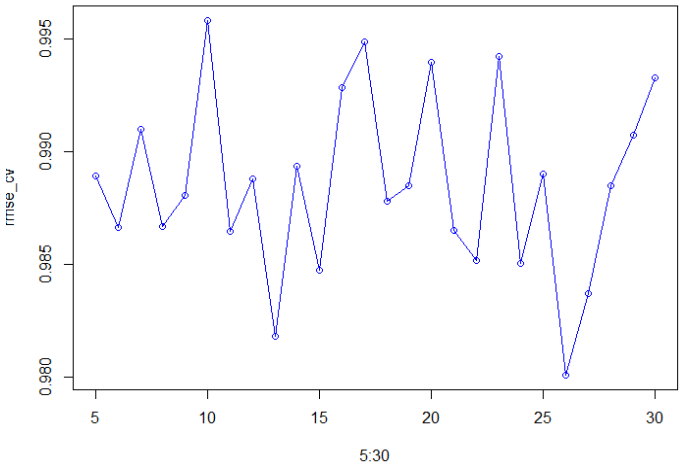
RMSE Plot for different number of iterations -



- Now we will k cross validation in which we decide the best k value based on the least RMSE.

```
#ID3 with k-fold cross-validation
rmse_cv=list()
for (k in 5:30){
  train_control=trainControl(method='cv',number=k)
  #here k is number of folds for cross validation
  model=train(Walc~.,data=mydata,method='rpart',trControl=train_control,metric='RMSE',maxdepth=5)
  rmse_cv[[length(rmse_cv)+1]]=model$results$RMSE[1]
  print(model)
}
print(rmse_cv)
plot(5:30,rmse_cv,type="o",col='blue')
#the optimal k is for which the rmse is least
k=which(rmse_cv==min(unlist(rmse_cv)))+5-1
```

RMSE plot for different values of k -



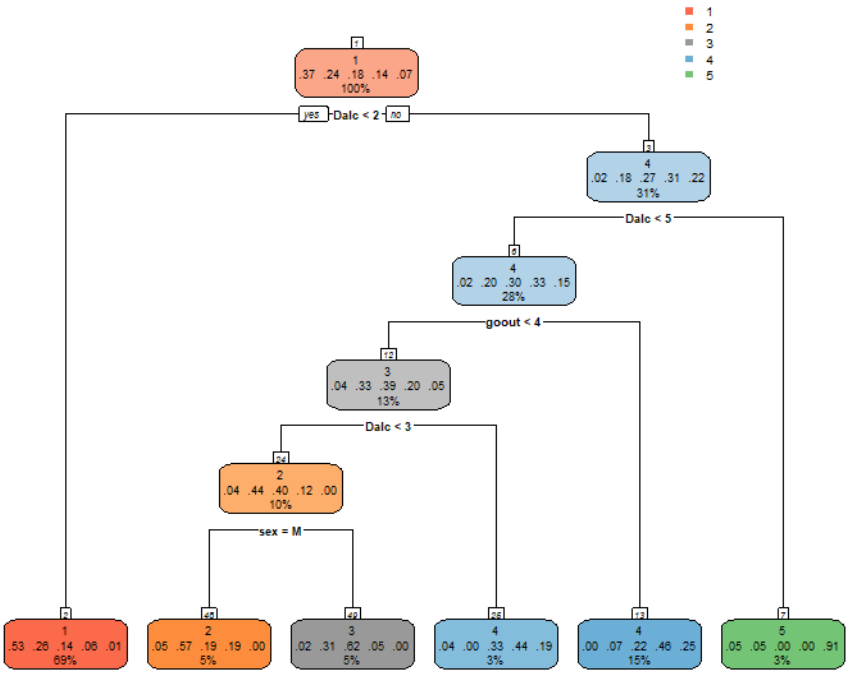
The best possible k value is -

k	26

- With best possible k we will find the decision tree and confusion matrix

```
#confusion matrix,accuracy,f1 score,recall,precision
model=rpart(walc~., data=mydata.train,xval=k, method="class")
rpart.plot(model, nn=TRUE)
pred <- predict(model,subset(mydata.test,select=-c(walc)),type="class")
cm=confusionMatrix(factor(pred,levels=1:5),factor(mydata.test$walc,levels=1:5))
print(cm)

#confusion matrix
print(paste('the confusion matrix:'))
print(cm$table)
```

Decision tree Will be -



Confusion matrix -

```
> print(paste('the confusion matrix:'))
[1] "the confusion matrix:"
> print(cm$table)
          Reference
Prediction  1  2  3  4  5
         1 85 27 24  8  2
         2  1  3  6  2  1
         3  0  2  5  0  0
         4  1  5 13 15  4
         5  0  0  0  0  4
```

For the statistical measures -

```
#accuracy
print(paste('the overall accuracy:',cm$overall['Accuracy']*100))

#balanced accuracy
print(paste('Balanced Accuracy by class:'))
print(cm$byClass[,'Balanced Accuracy'])

#recall
print(paste('Recall by class:'))
print(cm$byClass[,'Recall'])

#precision
print(paste('Precision by class:'))
print(cm$byClass[,'Precision'])

#F1 score
print(paste('F1 score by class:'))
print(cm$byClass[,'F1'])
```

We will get Accuracy,precision,recall,F1 score as -

```
> print(paste('the overall accuracy:',cm$overall['Accuracy']*100))
[1] "the overall accuracy: 53.8461538461538"
> #balanced accuracy
> print(paste('Balanced Accuracy by class:'))
[1] "Balanced Accuracy by class:"
> print(cm$byClass[,'Balanced Accuracy'])
 Class: 1  Class: 2  Class: 3  Class: 4  Class: 5
0.7364396 0.5113008 0.5458333 0.7371585 0.6818182
> #recall
> print(paste('Recall by class:'))
[1] "Recall by class:"
> print(cm$byClass[,'Recall'])
  Class: 1   Class: 2   Class: 3   Class: 4   Class: 5
0.97701149 0.08108108 0.10416667 0.60000000 0.36363636
> #precision
> print(paste('Precision by class:'))
[1] "Precision by class:"
> print(cm$byClass[,'Precision'])
 Class: 1  Class: 2  Class: 3  Class: 4  Class: 5
0.5821918 0.2307692 0.7142857 0.3947368 1.0000000
> #F1 score
> print(paste('F1 score by class:'))
[1] "F1 score by class:"
> print(cm$byClass[,'F1'])
 Class: 1  Class: 2  Class: 3  Class: 4  Class: 5
0.7296137 0.1200000 0.1818182 0.4761905 0.5333333
```

Thank You